Android Network Management

CS7012 Management of Network and Distributed Network

Sarah Alzahrani                Jinwei Gu
Fengjiao Lv                    Xuejiao Liu
Simerpreet Singh Anand

Trinity College Dublin
2014-2014
**Table of Contents**

1. Motivation .............................................................................................................................. 3
2. Objectives ............................................................................................................................... 3
3. Literature Review ................................................................................................................... 4
   3.1 Existing SNMP Android Applications ............................................................................... 4
   3.2 Android Operating System ............................................................................................... 9
      3.2.1 Background ............................................................................................................... 9
      3.2.2 History ...................................................................................................................... 9
      3.2.3 System Architecture ............................................................................................... 10
      3.2.4 System Requirement ............................................................................................... 11
   3.3 Simple Network Management Protocol .......................................................................... 12
      3.3.1 SNMPv1 .................................................................................................................. 12
      3.3.2 SNMPv2 .................................................................................................................. 12
      3.3.3 SNMPv3 .................................................................................................................. 12
      3.3.4 SNMP&CMIP .......................................................................................................... 13
      3.3.5 SNMP Libraries ....................................................................................................... 13
4. Requirements for Prototype ..................................................................................................... 15
   4.1 Technical Requirements of the Prototype ......................................................................... 15
5. Architecture Design ................................................................................................................ 17
   5.1 Overview of SNMP Architecture .................................................................................... 17
   5.2 Simplistic SNMP Architecture representation .................................................................. 19
   5.3 SNMP Manager Design ................................................................................................... 19
   5.4 SNMP Agent Design ......................................................................................................... 20
6. Implementation ....................................................................................................................... 21
   6.1 Manager Component: ....................................................................................................... 21
   6.2 Agent Component .............................................................................................................. 23
7. Evaluation ............................................................................................................................... 26
8. Project Management ............................................................................................................... 28
9. Conclusion ............................................................................................................................... 29
References .................................................................................................................................. 30
Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SNMP Manager main interface [4]</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>SNMP Traffic Grapher [10]</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>ezNetScan [12]</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>Android System Architecture</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>the SNMP4J API Stack [22]</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>Java Classes of SNMP Manager</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>Protocol Data Unit.</td>
<td>18</td>
</tr>
<tr>
<td>8</td>
<td>Android SNMP Architecture</td>
<td>19</td>
</tr>
<tr>
<td>9</td>
<td>flow among the Java classes that make up the Manager.</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>classes that build up to make the SNMP Agent</td>
<td>20</td>
</tr>
<tr>
<td>11</td>
<td>SNMP main interface</td>
<td>21</td>
</tr>
<tr>
<td>12</td>
<td>Agent interface</td>
<td>23</td>
</tr>
<tr>
<td>13</td>
<td>Agent Sample Run</td>
<td>26</td>
</tr>
<tr>
<td>14</td>
<td>Manager Sample Run</td>
<td>26</td>
</tr>
</tbody>
</table>
1. Motivation

Today’s network is more flexible than stable, as it used to be in early days. Mobile devices like cell phones have connected places where digital data never reached before. And these devices can prove to be boon for technological gateway, in the field of the management of networks.

Our motivation for this paper is to study the scope of Android device, in network management. Per previous studies [1] [2] [3] [4], are more about monitoring and managing SNMP enabled network devices such as routers by using Android devices, but few study such as [5] has treated the Android device as managed object as whole.

In this paper, we have tried to scope the Android device in network management considering it as managed object, whose objects can be monitored and controlled. Having, Android as managed objects we then will be able to ‘get’ the real time object value and also ‘set’ to change the object value on run time.

2. Objectives

Objective for this paper first is to make the device manageable with remote manager. Secondly to find the scope where the managed objects can be used in the daily life. Third, to evaluate such a model if it can be good way to use Android in management of networks.

The first part can be done using the API’s of both the Android and SNMP. Second objective can be done with comparing the study and with previous studies and enhancing our product in terms of the roles and scope. Roles that are played by each managed objects. Identifying the real scope for this study is of real importance, since main objective of the study is to evaluate and justify the Android use in network management.

Scope could be unlimited, in a way, a phone can managed and controlled completely using a remote manager that may even include the option to let the Android phone owner to answer the call or not. However, this idea seems to be beyond the scope to justify third point.
3. Literature Review

3.1 Existing SNMP Android Applications

The use of smartphones and tablets has grown to a widespread level today. Android is the world’s most popular operating system for these devices. The openness of Android has made it the favorite platform for both consumers and developers [6]. As a result, the number of the applications in the android market has reached 1.4 million in February 2015 [7].

In Google Play which is the primary source for android apps, only few applications has targeted the network management area (SNMP in particular).

**SNMP MIB** browser is name of application that can be downloaded for free from google play. It developed by Zoho Corporation which was founded in 1996. This application allows the user to browse and view the MIB data of SNMP enabled network devices such as routers, switches and Desktops. Also it enables users to load any standard MIB and fetch values from Agents that located on the configured devices. It supports all the versions of SNMP. In the case of SNMPv3, the tool allows the user to configure credentials of several users to be used to communicate with specific agent. It also implements algorithms like SHA and MD5 for authentication and DES, 3DES, AES-128, AES 192, AES-256 encryption for SNMPv3 communication [8].

**SNMP Manager** is a free application that can be found also in google play. It has high installation rate in comparison to the other tools that targeted SNMP. It has simple interface (Figure 1) and basic features such as MIB browser, trap reception and trap transmission. It supports SNMPv3 [4].
MIB Explorer Android is a commercial MIB browser application that is designed for android devices. It has been claimed that this tool is the first App to support SNMP table browsing with a user friendly interface and maximum MIB utilization. It provides various features such as allowing the user to browse SNMP enabled devices using SNMPv1, SNMPv2 and SNMPv3. Also users by using their android devices are able to configure and monitor their routers, switches, servers, modems... etc. it support Trap reception as well [9].

Another free tool called ColdStart which is different from the above tools as it mainly focuses on the trap reception. It allows network administrators receive traps from the SNMP enabled network devices such as routers and switches on their android devices instantly. It does not require any prior configuration, users only need to set the trap destination. In this software communication between the android device and the ColdStar Trap service is done over SSL. The source code of this tool is available via their official website [1], [2].
**SNMP Traffic Grapher** is one of the basic SNMP free tool that has high installation rate. The main function of this software is to illustrate the download and upload data of SNMP from device on real time graph (Figure 2). Some conditions need to be met before using it. [10]

![SNMP Traffic Grapher](image)

*Figure 2 SNMP Traffic Grapher [10]*

**ezNetScan** and **ezNetScan+** are two version of the same product that developed by VR SOFTWARE. The latter is the commercial version and it has more advanced features. It designed for both network administrators and general users who are interested in the network management.

The software has many useful features such as Scan WIFI network and display a list of the connected users as well as their details, Send ping request to required device, Scan active TCP ports, Perform basic operations on SNMP enabled devices like collect information about the monitored devices (software and hardware details) [3],[11],[12].
**SNMP Trap Agent** is also a free application that can be found in Google Play. It designed for both personal or enterprise use. The application main purpose is to monitor various things in the Android device such as performance, location, usage... etc. it works as service in the background of the device and it is password protected to provide the necessary security. Also it is capable to send Trap regularly for example every 5 minutes or when there is a problem. By using this app users can check the information that would be include in the trap message such as device’s location, percentage of free memory, device ID, message written by the user and more. The main limitation of this tool is that the SNMP port should be open across the network [13], [14].

As reported above most of the applications have targeted the android device as manager that can monitor the SNMP enabled network devices while only few of them treated the android device as managed object.
In [5] Hidalgo and Gamess claimed that their work bring the first implementation of an SNMP agent to Android. Their agent supports SNMP version1 and version2. After developing the agent they perform experiments to study the SNMP traffic that can be handled by their agent to prove that it is realistic to integrate mobile Android devices in network management or not.

Two metrics which are response time and Reply Request Radio (RRR) were tested to evaluate the performance especially in stressed situation. In their tests various numbers of OID and GetRequest were sent to the agent. They perform their test on six different android devices. The results obtained from the experiment show that Android mobile devices have enough power to be integrated in a SNMP management system since they are able to handle large amount of SNMP request within reasonable period of time.

Their design of the evaluation tests is quite useful to make performance measurement. However, what if the phone become overflowed by receiving too many GetRequest? The paper failed to mention how to control overflowed requests and how to tolerate the fault. Another limitation is that their agent does not support SNMPv3 but they mentioned that would be their future work.
3.2 Android Operating System

3.2.1 Background

Nowadays, smart phone is becoming more and more popular and at the same time it plays a vital role in people's daily life. With the dramatic growth of smart phone, various kinds of mobile platform occur and the number of mobile application correspondingly increases.

Android, one of the world's most popular mobile platforms, it "powers hundreds of millions of mobile devices in more than 190 countries around the world and it's also the largest installed base of any mobile platform and growing fast". [6]

3.2.2 History

There are many different versions of Android Platforms. "Since these Android devices make our lives so sweet and wonderful, each Android version is named after a dessert." [15] At the same time, each release also add something new whether the new stuff is improving the performance or just opening up new horizon in games area.

The main evolution of Android version by API level as well as name is shown in the Table below.

<table>
<thead>
<tr>
<th>Version</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android 1.5</td>
<td>Cupcake</td>
</tr>
<tr>
<td>Android 1.6</td>
<td>Donut</td>
</tr>
<tr>
<td>Android 2.0</td>
<td>Eclair</td>
</tr>
<tr>
<td>Android 2.2</td>
<td>Froyo</td>
</tr>
<tr>
<td>Android 2.3</td>
<td>Gingerbread</td>
</tr>
<tr>
<td>Android 3.0</td>
<td>Honeycomb</td>
</tr>
<tr>
<td>Android 4.0</td>
<td>Ice Cream Sandwich</td>
</tr>
<tr>
<td>Android 4.1</td>
<td>Jelly Bean</td>
</tr>
<tr>
<td>Android 4.4</td>
<td>KitKat</td>
</tr>
<tr>
<td>Android 5.0</td>
<td>Lollipop</td>
</tr>
</tbody>
</table>
3.2.3 System Architecture

Android Operating System is architected in the form of a stack of software components. The stack includes applications, Linux Kernel, run-time environment, libraries, services and middleware. Furthermore, the stack is divided into 4 main layers. It is presented virtually in Figure 4.

Applications: Android usually is released together with a series of core application packages. The package contains email client, SMS message program, calendar, maps, browser, etc. All these applications are written by using JAVA.

Application Framework: This layer provides many higher-level services to applications in form of Java classes. Android developers could access to the APIs which are used by core application. The application framework simplifies the reuse of components and the reuse mechanism also enables users to replace components conveniently.
Libraries: These libraries can be used by different components of Android System. They use the Android application framework to supply services to developers.

Android Runtime: Android includes a core library which provides most functions of Java core library and a Virtual machine where the applications run in.

Linux Kernel: This layer provides basic system functions like storage management, process management, security, device management etc. It provides an execution environment for multiple processes which usually execute concurrently. Linux Kernel also acts as an abstract layer between the hardware and software stack.

3.2.4 System Requirement

Android development need uses some technologies.

- **Eclipse**
  It is a significant IDE (Integrated Development Environment) for Android Application Development.

- **Android Developer Tools (ADT)**
  This is a plug-in of Eclipse. In order to create the Android Project, we need first install the plug-in in Eclipse.

- **Software Development Kit (SDK)**
  SDK is a set of development tools which is established for particular software packages (such as software framework, hardware platform, operating system) by software engineers.

In Android development, SDK provides libraries and other necessary development tools for Android developers. It could be simply understood as a development kit set. If you do not use Eclipse as your development software, you can just download SDK to develop without installing ADT.
3.3 Simple Network Management Protocol

3.3.1 SNMPv1

In the late 1980s, the Simple Gateway Management Protocol (SGMP), the predecessor of SNMP, was used to manage the bridges and gateway servers [16]. In 1988 Jeff Case, James Davin, Mark S. Fedor and Martin Schoffstall issued the Simple Network Management Protocol (SNMP), which is a low-overhead, easily implemented approach to manage network resources such as routers, servers, workstations and wiring concentrators [17]. The initial version of SNMP, called SNMPv1, has become the most widely-used tool for network management.

However, Frank Henderson researched and mentioned that there are several limitations of SNMPv1 [18]. The most significant drawback is SNMPv1’s security capabilities are minimal. The security strategy of SNMPv1 is vulnerable, and recovery mechanism could bring down the backbone network. In addition, another problem is the ability to sift network alarms to determine failed devices. In network system, the real issue is fault correlation. Nevertheless, all alarms are treated equally and difficult to find the real problem in SNMPv1.

3.3.2 SNMPv2

In 1996 January, SNMPv2 was designed to address the shortcomings in SNMPv1. SNMPv2 provides two advanced mechanism: polling and security. Instead of SNMPv1’s weak security strategy, SNMPv2 employs a three-layer security strategy: encryption, authentication and authorization. Furthermore, SNMPv2 provides better support for decentralized network management architecture because introducing two new features: an inform command and a manager-to-manager MIB [19].

3.3.3 SNMPv3

In 1996 September, taking account into both SNMPv1 and SNMPv2 are lacking security features, SNMPv3 was designed to define the principal security facilities: authentication, privacy, and access control [20]. In addition, SNMPv3 includes the functionality of SNMPv2. Authentication means that agents can verify a command id from an authorized manager and the contents have not been modified. Managers and agents encrypting messages to support implementation of privacy. Finally, the access control facility provides different levels of access to different managers.
3.3.4 SNMP&CMIP

Based on ISO model Common Management Information Protocol (CMIP) was designed as a contender, which unify standard for network management, operations and administration. Ray Hunt researched and compared CMIP and SNMP and indicated differences [21]. CMIP is object-oriented, therefore it is efficient and flexible. However, it requires significantly more memory and processing overhead. As for limited memory devices, SNMP is widely used in LANs and interconnected networks to manage hardware. Thus, CMIP is more popular for telecommunications network management, SNMP is more suitable to be employed in portable handheld devices network management.

3.3.5 SNMP Libraries

With the development of SNMP protocol, there are several popular SNMP libraries for different programming languages.

- Net-SNMP: Net-SNMP is a free, open source SNMP implementation, formerly called UCD-SNMP. It includes multiple agent and source code management tools, support for multiple expansion mode. Net-SNMP library supports C and Perl language.
- PySNMP: PySnmp is a cross-platform, based on Python SNMP implementation. Besides the basic libraries, PySnmp provides a set of pure-Python tools. Those tools mimic the interface and behaviour of Net-SNMP utilities.
- SharpSNMP & SnmpSharpNet: Both are free open SNMP library for C# language.

This project based on Android platform, therefore, a SNMP library for JAVA will be used. However, there are many existing JAVA libraries, such as SNMP4J, Network Ferret (NF), joeSNMP (name change from jSNMP) and so on.

By comparison with other libraries, in this project we will use the SNMP4j library. Which is object-oriented, and it supports in the form of the command line to manage and respond. Taking into account reliability and availability, SNMP4j APIs support synchronous and asynchronous requests and multithreading. In addition, SNMP4j can tolerate network congestion. In terms of security, the API of SNMP4j provide multiple Secure Hash Algorithm and Message Digest Algorithm. [22]
Figure 5 the SNMP4J API Stack [22]
4. Requirements for Prototype

Before we start to develop distributed system, that communicates using the SNMP, we identify the various components of the basic SNMP architecture that fits in our distributed systems design. Following are the basic components for SNMP architecture (discussed below in Technical Architecture):

1- SNMP Manager
2- SNMP Agent
3- Managed Object

We tend to monitor and manage the android mobile devices from application at the center in our design. SNMP Manager is the standalone network program which retrieves information from each Managed Objects (identified as the distributed components). Android devices are the managed objects in our design, SNMP Manager is required to retrieve information from each distributed component in the system and process that information. SNMP Agents helps to fetch information from each managed object, pack it and send it to the respective manager.

Now that we are clear with the components of SNMP architecture, we will look into the technical requirements of the above components.

4.1 Technical Requirements of the Prototype

4.1.1 SNMP Manager:

Primary objective of Manager is to have to access to the data from managed objects (Android devices), which it needs to monitor and manage using the SNMP communication. Our SNMP Manager is the Java standalone program, enabled with SNMP Get(), Set() methods and listening to SNMP Trap messages from Agents. Manager need to have all PDU’s from Agents, and process the information and monitor/manage it.
Above figure, represents the Java Classes that makes the SNMP Manager. MainGUI class that shows the list of Android devices in range and Manager Class that launch the SNMP Manager and GUI. AndroidDevice class used with SNMPMessenger class use GET(), SET() methods and other important class called MultiThreadTrapReceiver to listen to Agents Trap messages.

4.1.2 SNMP Agent

SNMP Agent is close component to the managed object, which sends over the PDU’s to the Manager. PDU (Protocol Data Unit) is the data set which states the current state of android device managed objects. SNMP Manager communicates with Agent to fetch real time data of objects it manages, when SNMP sends the Get() request the Agent replies with PDU’s.

SNMP Agent in our design is much similar to General Agent in SNMP framework, which runs to provide basic SNMP functionalities.

- Transmission protocol definition: UDP is the one we chose as of now.
- Create instance of SNMP class.
- Start SNMP listener.
- Create protocol data unit (PDU).
- Reply to GET() request from SNMP Manager.
- Trap message communication with SNMP Manager.

4.1.3 Managed Objects

Managed Objects are the Android device objects which have some property to be seen or monitored or managed. Android device is the distributed component in our design, whose properties being tracked by SNMP Manager via SNMP Agent.
5. Architecture Design

5.1 Overview of SNMP Architecture

The SNMP architecture has got three components:

- Manager (Java network program using SNMP library): The Manager is responsible for communicating with, and managing, android devices that implement SNMP Agents.
- Agents (Java network program using SNMP library): The Agents reside in the android device and they expose information to the Manager.
- Managed Objects: Objects (i.e. Managed Objects) to be managed by the Agent running within the android device. Objects are real time properties of the Android mobile phone device, whose values are actually measurable and identifiable.

Very basic functioning of the SNMP architecture is accessing the data. In SNMP data is accessed from managed objects by Agent and passed on to the Manager; MIB comes into the picture here. Right time to describe it, MIB (Management Information Base) is a set of information/data organized hierarchically. Now since, MIB store the information, SNMP needs a way to identify the objects in the MIB. Just to show the complexity of data stored in MIB, MIB can be used to store multilevel of information which relates to each other. Each object entry in the MIB has got the ‘unique’ ID.

Now, to access the objects from MIB, OID is used in SNMP architecture. OID stands for Object Identifiers, which uniquely identifies the managed objects in the MIB hierarchy. Since SNMP v1, the data can be transported using different transport protocols. In our design we are transporting the data using the UDP. In our design, SNMP Manager and SNMP Agent are talking to each other on local network.

The communication between SNMP Managers and SNMP Agents in our architecture is pretty much straight forward:

- GET method is called by SNMP Manager to SNMP Agent (who resides in Android Device itself) to request data values defined by a PDU which takes parameters (such as transmission protocol, PDU data, SNMP object). The Agent will respond with the requested values. The SNMP architecture for Android device uses GET operations to retrieve data from the MIB’s of Agents.
- SET method is again called by SNMP Manager to SNMP Agent. In our SNMP Architecture we are using SET operation since the Manager is responsible for both monitoring and managing. As of now when it comes to managing the Android from Manager GUI, it we can send request to switch on/off the Bluetooth, however, more
such properties from Android can be used and more can be identified to be managed, just like the Battery usage is one.

- Trap message can be used to send the important messages from Agent to the Manager, to alert them of important events in managed device.

```
public void sendGetRequest(OID oid, final SNMPResponseListener responseListener) {
    try {
        // Create the PDU object
        PDU pdu = new PDU();
        pdu.add(new VariableBinding(new OID(oid)));
        pdu.setType(PDU.GET);
    } catch (Exception e) {
        // Handle exception
    }
}
```

The above snippet from Manager Class code shows, where a Get request is prepared, sent from Manager to the Agent.

<table>
<thead>
<tr>
<th>PDU Type Value</th>
<th>PDU Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>GetRequest-PDU</td>
</tr>
<tr>
<td>1</td>
<td>GetNextRequest-PDU</td>
</tr>
<tr>
<td>2</td>
<td>Response-PDU</td>
</tr>
<tr>
<td>3</td>
<td>SetRequest-PDU</td>
</tr>
<tr>
<td>4</td>
<td>Obsolete, not used (this was the old Trap-PDU in SNMPv1)</td>
</tr>
<tr>
<td>5</td>
<td>GetBulkRequest-PDU (has its own format, see below)</td>
</tr>
<tr>
<td>6</td>
<td>InformRequest-PDU</td>
</tr>
<tr>
<td>7</td>
<td>Trapv2-PDU</td>
</tr>
<tr>
<td>8</td>
<td>Report-PDU</td>
</tr>
</tbody>
</table>

Figure 7 Protocol Data Unit.
5.2 Simplistic SNMP Architecture representation.

The figure above represents basic architecture view to understand SNMP view of our design. On the left is the SNMP Manager and SNMP Agent lies in the Android device where also lies our Managed Objects which are shown as Android symbol. Later in this section you will see the technical architecture.

5.3 SNMP Manager Design

SNMP Manager is developed using the SNMP4j library, incorporating SNMP version 1 and version3. More detailed information will be followed in implementation section, meanwhile adding a snippet of the Java classes used to make up the SNMP Manager and their flow in Android management SNMP architecture.
5.4 SNMP Agent Design

SNMP Agent is developed using the SNMP4j library again, and is substituted in the android device. SNMP agent are used to store the required information in MIB (Managed Information Base), which are accessed using the OID’s, and is passed on to the SNMP Manager.

Below is the set of Java classes with SNMP and Android library developed to run as SNMP agent, more detailed explanation will be followed up in later ‘implementation’ section.

Agent is also used to send the Trap message back to the SNMP manager, subpart of the AgentService class, which sends the data back to the Manager.

Figure 10 classes that build up to make the SNMP Agent.
6. Implementation

Our android management system consists of two software components: a Manager that is used to monitor as well as manage Android devices and an Agent that has MIB. However, each component was implemented and tested separately. The manger is a desktop application while the agent is an android application. Continuously during the implementation phase both components were tested to make sure that they can communicate together.

6.1 Manager Component:

The manger component implements SNMP polling (Get and Set) and SNMP Trap. The following screenshots show the functionalities of the manager. It can get different information about the android device that is connected to it such as GPS status, Bluetooth status, Battery status (if the phone is charging or not), Model name, RAM level, CPU level and version name. Also, manger can send messages to the agent and then agent will send a response back which is an OID pulse the message that was sent. Moreover, manager can turn on the Bluetooth and Wi-Fi of the android device.

![Figure 11 SNMP main interface](image)
To start monitoring the android devices and send messages to it, user need to choose the SNMP version and type the IP address of the target android device. After that, if the user click on one of the labels, the monitoring information will be shown besides the label. Note: It should be buttons not labels but with the time constraints we decided to delay the task of improving our design.

The following samples code illustrate SNMP Set, Get and Trap that have applied in the project.

- SNMP set request to turn the android device Bluetooth on:

```java
class MyAgent {
    public void turnOnBluetooth() {
        snmpMessenger.sendSetRequest(new OID(new int[] {1,3,6,1,4,1,12345,1,3,4}), "1", new SNMPResponseListener() {
            public void onResponseReceived(Vector? extends VariableBinding? variableBinding) {
                setBluetoothStatus(variableBinding.get(0).toString().equalsIgnoreCase("on") ? "on" : "off");
            }
        });
    }
}
```

- SNMP get request to retrieve the user CPU time from the device.

```java
class MyAgent {
    public void updateCpuLevel() {
        snmpMessenger.sendGetRequest(new OID(new int[] {1,3,6,1,4,1,12345,11,9,0}), new SNMPResponseListener() {
            public void onResponseReceived(Vector? extends VariableBinding? variableBinding) {
                setCpuLevel(variableBinding.get(0).toString() + "%");
            }
        });
    }
}
```

- SNMP Trap which will listen for traps that is coming from the agent.

```java
class MyAgent {
    private void init() {
        try {
            ThreadPool = ThreadPool.create("Trap", 6);
            dispatcher = new MultiThreadedMessageDispatcher(ThreadPool, new MessageDispatcherImpl());
            dispatcher.setresponseData("snmp://listenAddress", udp://"ip"/"+port");
            // listen port
            TransportMapping transportnull; // listen port
            transport = new DefaultUdpTransportMapping((UdpAddress) listenAddress);
            System.out.println(transport.toString());
            snmp = new Snmp(dispatcher, transport);
            snmp.getMessageDispatcher().addListener(new MvPv2());
            snmp.listen();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

- Send messages to the agent

```java
class MyAgent {
    public void sendMessageToAgent(String message) {
        snmpMessenger.sendMessage(new OID(new int[] {1,3,6,1,4,1,12345,1,4,1,0}), message, new SNMPResponseListener() {
            public void onResponseReceived(Vector? extends VariableBinding? variableBinding) {
                System.out.println("response: " + variableBinding);
            }
        });
    }
}
```
6.2 Agent Component

There are four main classes in the agent implementation on Android.

**Activity Class:**
The Activity class controls the only one layout of Agent on Android. What the activity does is invoking the service in onCreate method and the most important function of it is demonstrating the message and requests which are sent from the Manager (PC). We also set a button on the layout to simulate dangerous condition, when you press the button, button listener will invoke the test Trap function.

In reality, the dangerous conditions usually present that System state has changed, Network status has changed, Disk state has changed, etc.

![Agent interface](image)

**AgentService class**
This class provides 5 basic functions:

1. When a new manager detects an Android device and the port number, the agent will add the manager into the list of registered clients. (Add client)

2. When a manager exits the system or the manager loses connection, the response message is also out of time, the manager will be removed out from the list of registered clients. (Delete client)

3. Agent sets the value and responds to the manager. (Set value)
4. If there is now in dangerous condition, the Agent will send the trap message to the manager. (Send trap) In this experiment, we use the Trap test OID {1,3,6,1,4,1,12345,100,0,99}.

5. If the agent finish manager's task, it will send response message to manager. (Respond to client)

Besides these, this class also provides basic SNMP functions:

1. Using DefaultUdpTransportMapping() to define the UDP transmit protocol in management system.

   ```java
   TransportMapping transport = new DefaultUdpTransportMapping(
       new UdpAddress("0.0.0.0" + SNMP_PORT));
   ```

2. Instantiating a new SNMP object.

3. Invoking an agent SNMP listener.

4. Processing PDU. According to different types of PDU - "GET""GETNEXT""SET", the Agent processes different messages.

5. Responding to manager's requests.

   ```java
   command.setType(PDU.RESPONSE);
   CommunityTarget target = new CommunityTarget();
   try {
       snmp.send(command, target);
   } catch (IOException e) {
       e.printStackTrace();
   }
   ```

As shown in the code above, this is the method how agent responds to manager's request. Target represents the remote device or remote instance, PDU represents the communication data between manager and Target, and SNMP represents the executor of management function which actually does the data transmission.

The third class is **system information** which mainly takes the charge of getting manager object's information. The agent monitors 9 main features of android device which are Device Model, Android Version, up time/ battery usage time, Battery Status, Battery Level, GPS status, Wi-Fi status, App information and Bluetooth status.

In the following code for example, agent could get the Bluetooth status of the monitored device.
Agent can also set some status of device: Turn on/off Bluetooth Technology and Turn on/off Wi-Fi connection.

As shown in the following code, Agent turns on the Wi-Fi connection on the Android Device.

```java
private void setWiFion()
{
    VariableBinding Vbind;
    OID oid = (OID)MIBtree.OPEN_WIFI_OID.clone();
    Wi-FiManager wifiManager = (WifiManager) context.getSystemService(Context.WIFI_SERVICE);
    if (!wifiManager.isWifiEnabled()) {
        wifiManager.setWifiEnabled(true);
    }
    Vbind = new VariableBinding(oid, new Integer32(1));
    MIB.set(Vbind);
}
```

The last class is **MIB** which defines all OID we used in the agent system.
"MIB stands for Management Information Base and is a collection of information organized hierarchically. These are accessed using the protocol - SNMP. OIDs (Object Identifiers uniquely identify) managed objects in a MIB hierarchy. This can be depicted as a tree, the levels of which are assigned by different organizations." [1]

```java
public static final OID SYS_ANDROID_VERSION_OID = new OID(new int[] {1,3,6,1,4,1,12345,1,1,1});
public static final OID SYS_UPTIME_OID = new OID(new int[] {1,3,6,1,4,1,12345,1,1,3});
```

Above are the samples of OIDs. These numbers are the ones used in PRTG when setting up custom sensors, in order to access some appropriate elements of the android device which is monitored by the agent. "OIDs are generally provided by hardware manufacturers or it can be found in OID repositories, where respective OIDs and the collections of MIB trees can be assessed."

*[Note: we have not started implementation from scratch. We get some ideas from an open source implementations for SNMP manager and SNMP agent [24]. The following link is the GitHub repository of our project: https://github.com/fengjiao/CS7012SNMP]*
7. Evaluation

Our goal for this project is to prove how successful the android platform in network management is and if it is good idea or not to use the android devices in management.

Most of the existing application on google paly used the phone as manager. However, based on that it has been decided to monitor and manage the android device rather than using it as a manger. This project has started with implementing basics functionalities that allows the manger to monitor the device status, send messages to the android device and receive traps from it.

Sample Run for the project:

1- Set, Get: in the following screenshots our manger sent three messages to the android device (Hello, hello world, your Bluetooth is open) while agent received three set requests from the manger which are the manger’s messages. Also, it received a get request for the Bluetooth status.
2- Trap

MultithreadTrapReceiver (1) [Java Application] C:\Program Files\Java\jre1.8.0_31\bin\javaw.exe (1 Ap
transport:org.snmp4j.transport.DefaultUdpTransportMapping@1459877
------------------------begin listening for trap information-----

In the above screenshot manager is listening for traps while the next screenshot shows the traps that the manager has received from the agent when an urgent condition as happened in the device.

transport:org.snmp4j.transport.DefaultUdpTransportMapping@1459877
------------------------begin listening for trap information-----
1.3.6.1.2.1.6.8:send string 1
1.3.6.1.2.1.1.2.0:send string 2
8. Project Management

Project Initiation

During this time the whole of group worked towards the finalization of the idea and focused on identifying the area of interest to each.

Project planning

Once the project area was identified, we worked towards the planning of the task initialization. With constant follow-up among the team mates on (via chat) we were very clear on our task and dependency on each other. While Fengjiao, Xuejiao completed the task of building the SNMP Manager and SNMP Agent and Jinwei helped both, during the process of development. While Sarah and Simer, studied SNMP architecture, and completed the task of precisely writing down project report, by understanding the work done by each member.

Project Execution and Performance

Execution throughout the project work remained very smooth and coordinated among the group members. With constant updates from all the team members, and remained in sync to complete the project milestone as decided during the initial phase of the project.

Project close

While the project concludes, we have SNMP enabled Agent and Manager Software, while the Agent is deployed as the Android program that talks to the internals of the Android program, and on the other end, SNMP Manager Java software, which request the data (Android properties) via Agent.
9. Conclusion

SNMP v1 is used to implement the SNMP architecture, and at the same time SNMP v3 is also been used to demonstrate the working with new version of the SNMP, additionally exploring the security part of it. SNMP3 was added explicitly to explore the security features of this protocol especially when messages are passed between the manager and agent.

In conclusion, from the existing android applications and our experience with this project, SNMP as a network management protocol has been successful in android platform. However, SNMP protocol is recognized as simple protocol but it is quite complicated to understand and implement in java.
References


[16] “RFC1157.”.

[17] “RFC3411.”.


